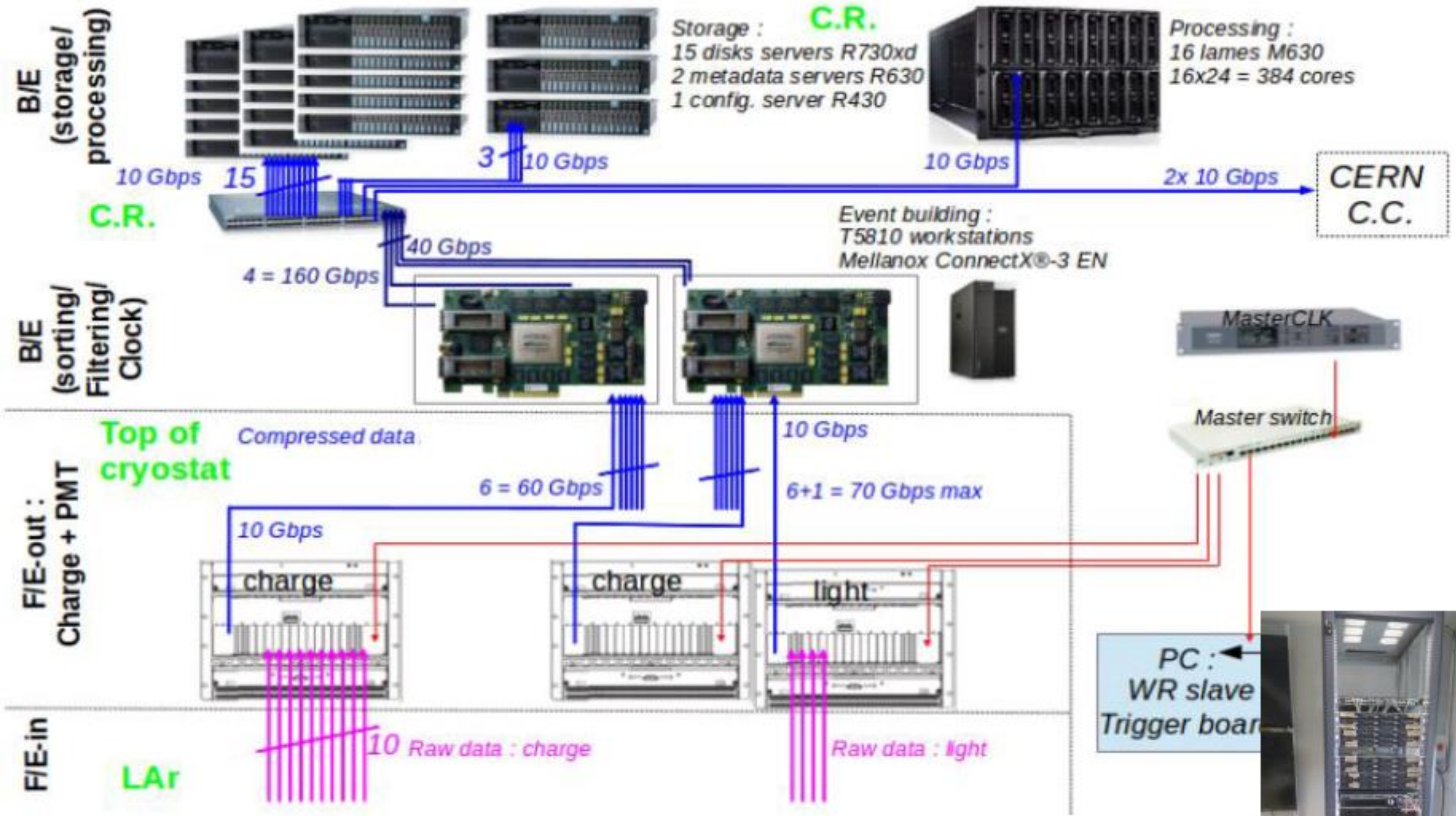
Developments and implementation of the WA105 6X6X6 online storage/processing on the 3x1x1 online storage and processing small scale test farm

Elisabetta Pennacchio, IPNL

WA105 SB meeting, November 9th 2016

DUNE meeting September 2016: **Online processing and storage facility of 6x6x6**

Online processing and storage facility: internal bandwidth 20 GB/s, 1 PB storage, 384 cores: key element for online analysis (removal of cosmics, purity, gain, events filtering)



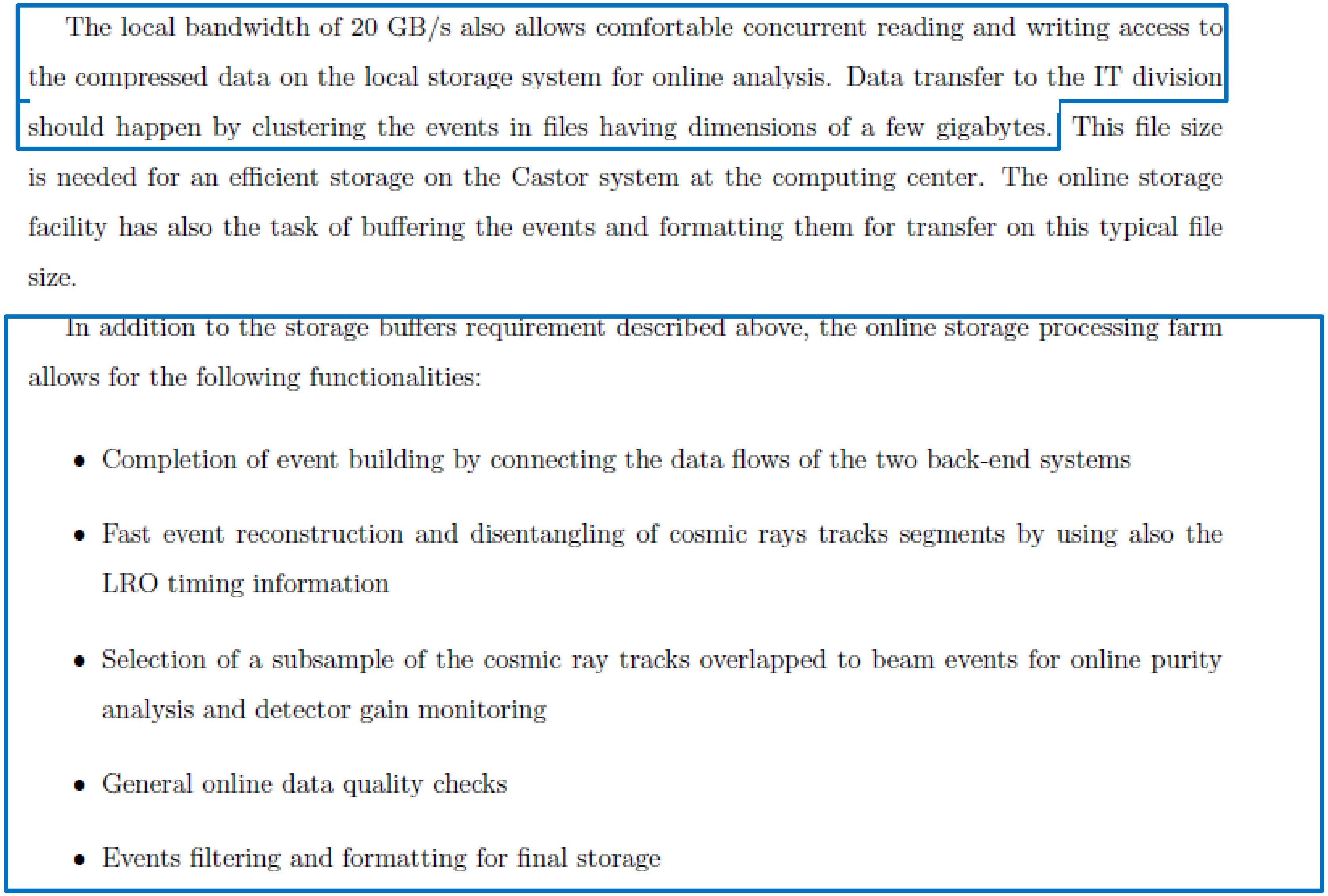
C.R. stands for Counting Room

Smaller scale test system implemented for the operation of the 3x1x1

Online storage/processing farm motivation :

SPSC report, April 2016

6x6x6

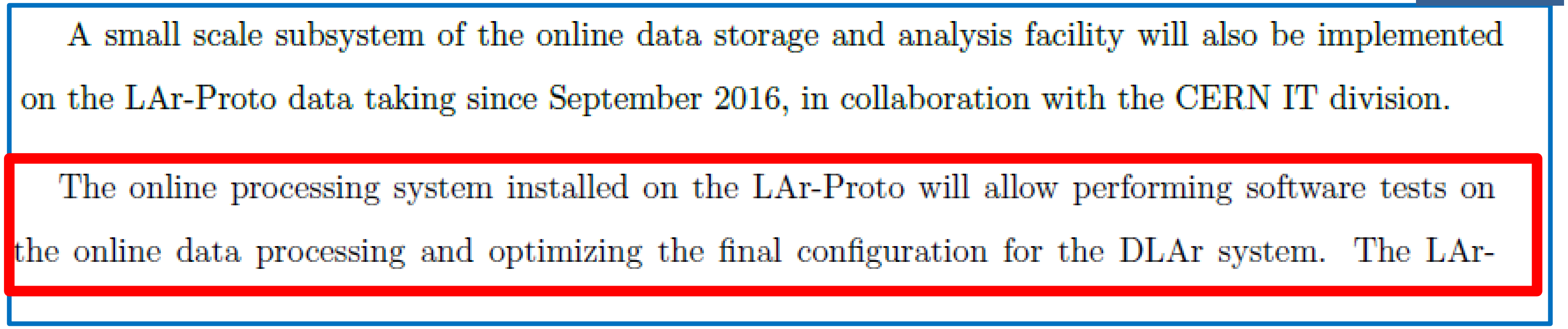


The local bandwidth of 20 GB/s also allows comfortable concurrent reading and writing access to the compressed data on the local storage system for online analysis. Data transfer to the IT division should happen by clustering the events in files having dimensions of a few gigabytes. This file size is needed for an efficient storage on the Castor system at the computing center. The online storage facility has also the task of buffering the events and formatting them for transfer on this typical file size.

In addition to the storage buffers requirement described above, the online storage processing farm allows for the following functionalities:

- Completion of event building by connecting the data flows of the two back-end systems
- Fast event reconstruction and disentangling of cosmic rays tracks segments by using also the LRO timing information
- Selection of a subsample of the cosmic ray tracks overlapped to beam events for online purity analysis and detector gain monitoring
- General online data quality checks
- Events filtering and formatting for final storage

3x1x1

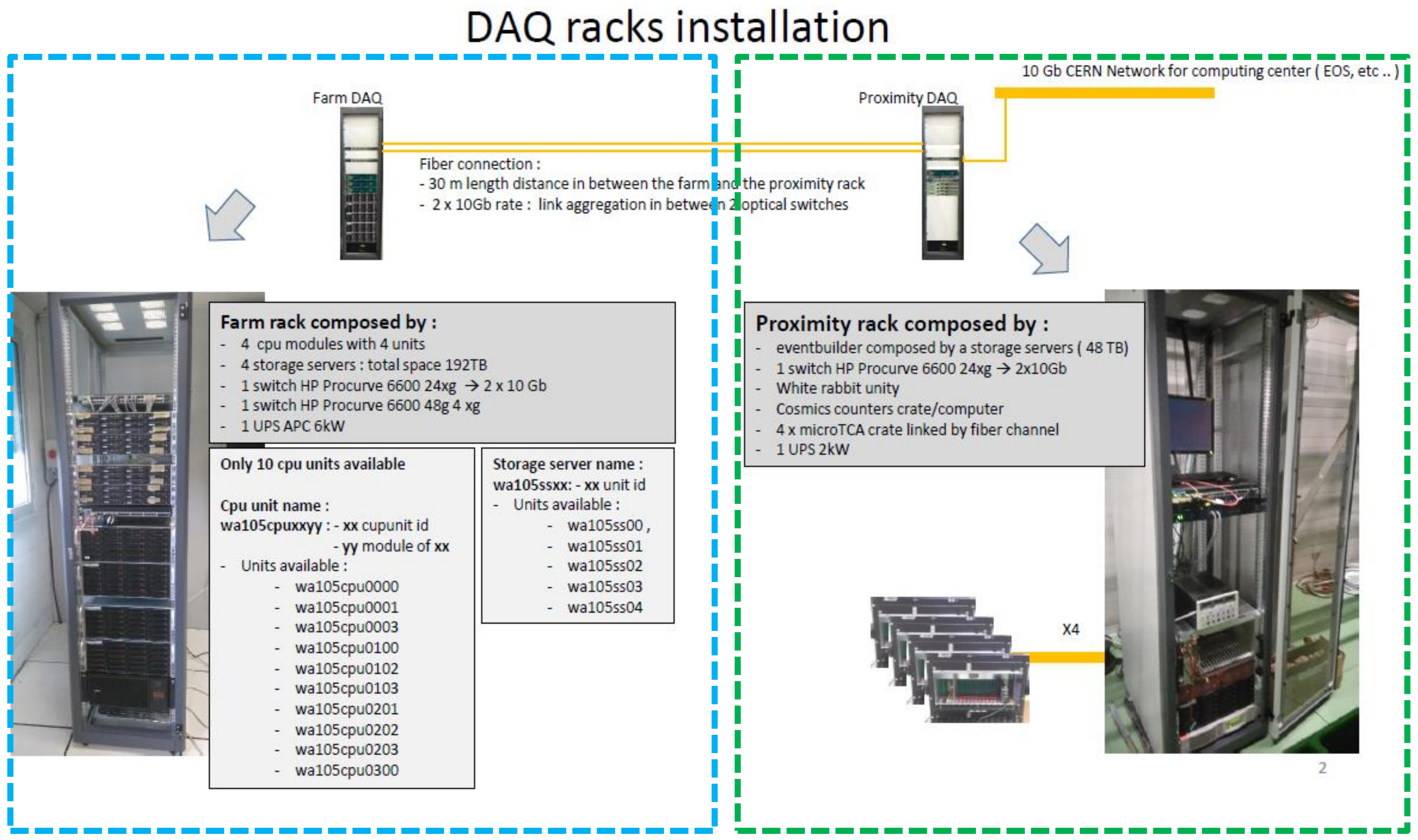

A small scale subsystem of the online data storage and analysis facility will also be implemented on the LAr-Proto data taking since September 2016, in collaboration with the CERN IT division.

The online processing system installed on the LAr-Proto will allow performing software tests on the online data processing and optimizing the final configuration for the DLAr system. The LAr-

A description of the hardware configuration of the farm has been provided during the 3x1x1 biweekly meeting of September 22nd
https://indico.fnal.gov/conferenceDisplay.py?confId=12944

The EOS system was selected months ago for the data storage system thanks to the extensive performance tests made by Denis Pugnere in order to grant high bandwidth concurrent write/read access (see TB slides) and it has been implemented also in the 3x1x1 smaller farm. EOS is a network distributed file system using a meta-data server (https://indico.fnal.gov/conferenceDisplay.py?confId=12347)

→ This presentation aims to show how the tasks of the online processing are implemented in the smaller scale farm available now and how this processing is organized for the 3x1x1



DAQ racks installation

Fiber connection :
- 30 m length distance in between the farm and the proximity rack
- 2 x 10Gb rate : link aggregation in between 2 optical switches

Farm rack composed by :
- 4 cpu modules with 4 units
- 4 storage servers : total space 192TB
- 1 switch HP Procurve 6600 24xg → 2 x 10 Gb
- 1 switch HP Procurve 6600 48g 4 xg
- 1 UPS APC 6kW

Only 10 cpu units available

Cpu unit name :
wa105cpuxxyy : - **xx** cupunit id
- **yy** module of **xx**
- Units available :
  - wa105cpu0000
  - wa105cpu0001
  - wa105cpu0003
  - wa105cpu0100
  - wa105cpu0102
  - wa105cpu0103
  - wa105cpu0201
  - wa105cpu0202
  - wa105cpu0203
  - wa105cpu0300

Storage server name :
wa105ssxx: - xx unit id
- Units available :
  - wa105ss00 ,
  - wa105ss01
  - wa105ss02
  - wa105ss03
  - wa105ss04

Proximity rack composed by :
- eventbuilder composed by a storage servers (48 TB)
- 1 switch HP Procurve 6600 24xg → 2x10Gb
- White rabbit unity
- Cosmics counters crate/computer
- 4 x microTCA crate linked by fiber channel
- 1 UPS 2kW

farm rack

proximity rack (event builder)

https://indico.fnal.gov/conferenceDisplay.py?confId=12944

Data flow

Binary files are written by the DAQ in the storage server of the proximity rack:
each file is composed by 335 events → 1GB/file (optimal file size for storage systems) not compressed

each run can be composed by several files (this number is not fixed):

run1-seq0.dat
run1-seq1.dat

run2-seq0.dat
run2-seq1.dat
run2-seq2.dat
run2-seq3.dat
run2-seq4.dat

The automatic online data processing includes these 3 steps (not in strict time order):

1) As soon as a data file is produced, it is copied to the EOS storage area of the farm, a script to run reconstruction is automatically generated and submitted to the batch system
2) Results from reconstruction (root files) are also stored in the storage area and analyzed to evaluate purity and gain, to monitor the behavior of the detector in time (*online analysis*)
3) The binary data files are also copied to the central CERN EOS, where they are available to the users for offline analysis

Note: Beyond the processing of the 3x1x1, which produces a very small data flow, the entire farm system is fundamental in "mock data challenges" and various tests in order to study and optimize the design of the final high rate system for the 6x6x6

Preliminary remarks

Before discussing how these steps have been implemented, it is useful to recall some main characteristics of the system (see also picture in slide 5):

- Event builder machine storage space: 48 TB
- Online storage/processing farm storage size: 192 TB
- Filesystem of the online storage system: EOS
- Protocol to copy data from Event Builder to the online storage system : xrootd
- Resources manager for the batch system: TORQUE
- 7 cpu units → 112 processors, possibility of having up to 112 jobs running simultaneously (job sequential monocore)

→ Reconstruction software installed

- WA105Soft (revision 377) and related libraries (root 5.34.23   xrootd 4.0.4, same versions installed at CCIN2P3 and on lxplus)
- The farm is foreseen for fast reconstruction of the raw data and purity and gain online measurement → only the code related to reconstruction has been installed (the code needed for generation of Monte Carlo events is not available)
- To validate this installation some Monte Carlo events (produced at CCIN2P3) have been processed   (reconstruction+benchmark) on the farm and on the CCIN2P3 computer center, producing identical results

→ Accounts on the online farm:

- shift → used by people on shift, to run the daq, the event display and monitor results
- prod → to maintain the automatic data processing machinery: scripts for file transfers, batch processing, copy to EOS
- evtbd → daq account for the event builder software maintenance

No personal account are foreseen on the farm
The account working environment is already finalized for the prod account
The configuration of the shift account is under study

→ To setup and test the data processing, 2 emulated data binary files produced by Slavic in the DAQ simulation tests in Lyon have been used

File 1: 335 events, compressed 37MB
File 2 335 events, not compressed 1GB

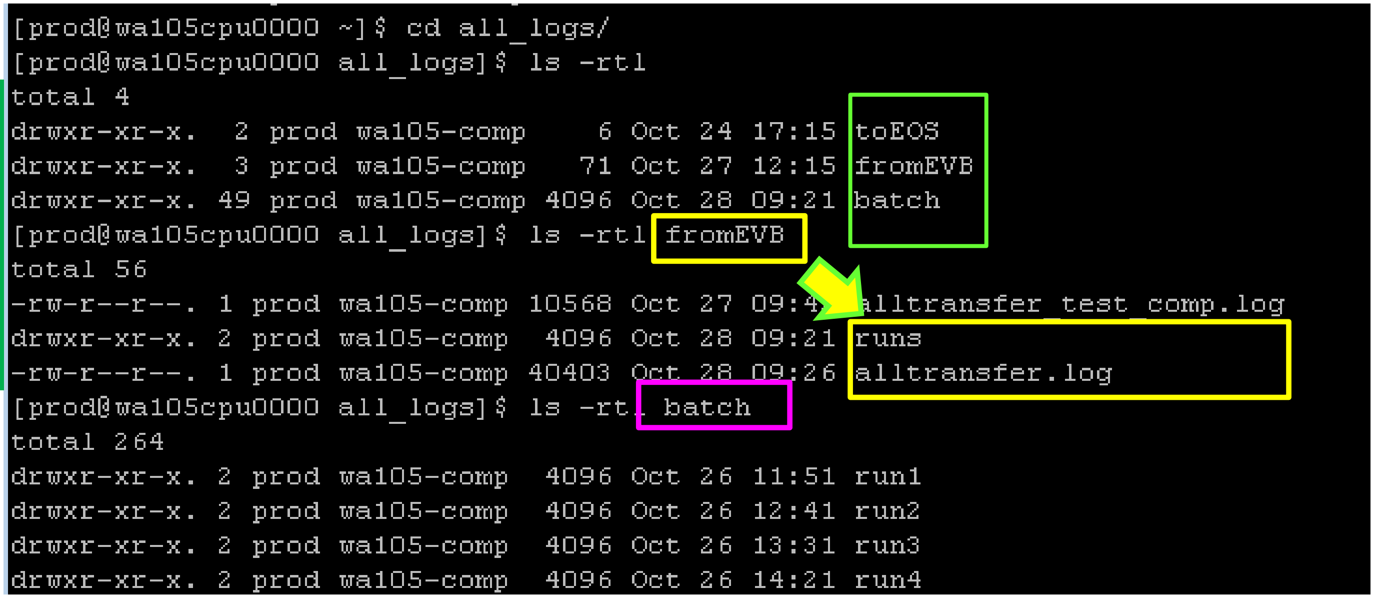
→ The configuration of the storage system is under finalization (local EOS installation) :
Standard UNIX commands to access, move, copy files are not the ones used under EOS
A temporary version of the processing/storage machinery scripts was developed and tested while EOS was not yet operational → all these scripts used on the farm to organize the data processing will have to be now to be re-adapted in the part of data access to be EOS compliant.

These changes, will have to be carefully tested, and will requires some times : It will not only be matter of modifying a couple of lines of code. In the following slides the red star ★ will show where changes are still needed

Directories in the production account

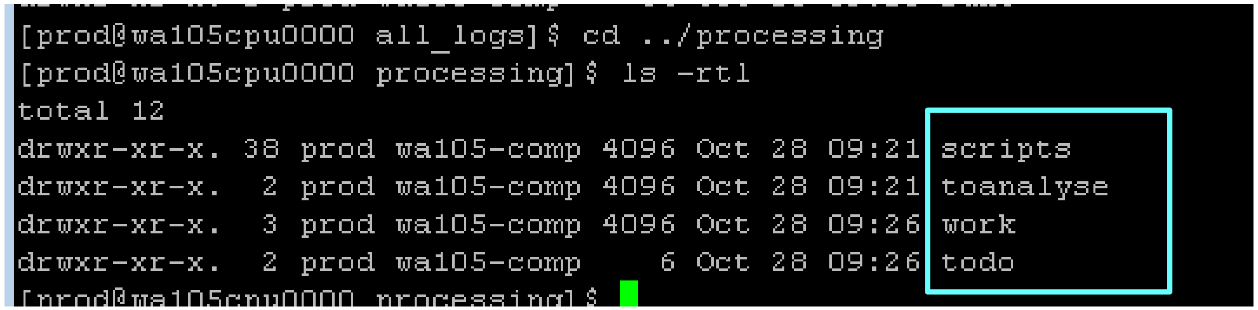
2 main directories:

all_logs: automatic processing logs/ bookkeeping



```
[prod@wa105cpu0000 ~]$ cd all_logs/
[prod@wa105cpu0000 all_logs]$ ls -rtl
total 4
drwxr-xr-x.  2 prod wa105-comp    6 Oct 24 17:15 toEOS
drwxr-xr-x.  3 prod wa105-comp   71 Oct 27 12:15 fromEVB
drwxr-xr-x. 49 prod wa105-comp 4096 Oct 28 09:21 batch
[prod@wa105cpu0000 all_logs]$ ls -rtl fromEVB
total 56
-rw-r--r--. 1 prod wa105-comp 10568 Oct 27 09:4  alltransfer_test_comp.log
drwxr-xr-x. 2 prod wa105-comp  4096 Oct 28 09:21 runs
-rw-r--r--. 1 prod wa105-comp 40403 Oct 28 09:26 alltransfer.log
[prod@wa105cpu0000 all_logs]$ ls -rtl batch
total 264
drwxr-xr-x. 2 prod wa105-comp  4096 Oct 26 11:51 run1
drwxr-xr-x. 2 prod wa105-comp  4096 Oct 26 12:41 run2
drwxr-xr-x. 2 prod wa105-comp  4096 Oct 26 13:31 run3
drwxr-xr-x. 2 prod wa105-comp  4096 Oct 26 14:21 run4
```

processing: processing scripts machinery/ bookkeeping



```
[prod@wa105cpu0000 all_logs]$ cd ../processing
[prod@wa105cpu0000 processing]$ ls -rtl
total 12
drwxr-xr-x. 38 prod wa105-comp 4096 Oct 28 09:21 scripts
drwxr-xr-x.  2 prod wa105-comp 4096 Oct 28 09:21 toanalyse
drwxr-xr-x.  3 prod wa105-comp 4096 Oct 28 09:26 work
drwxr-xr-x.  2 prod wa105-comp    6 Oct 28 09:26 todo
[prod@wa105cpu0000 processing]$
```

Now we will go through the 3 points of slide 6:

1) As soon as a data file is produced, it is copied to the EOS storage area of the farm, a script to run reconstruction is automatically generated and submitted to the batch system

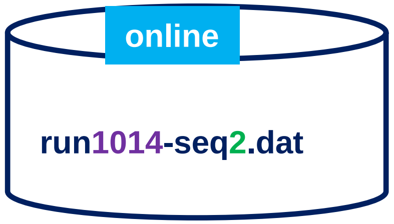
2) Results from reconstruction (root files) are also stored in the storage area and analyzed to evaluate purity and gain, to monitor the behavior of the detector in time (*online analysis*)

3) The binary raw data files are also copied to the central CERN EOS, where they are available to the users for offline analysis

Data Processing (point 1)

1) As soon as a new data file is produced, it is copied to the EOS storage area of the farm, a script to run reconstruction is automatically generated and submitted to the batch system

A binary file become available on the storage area of the online machine:
ex *run1-seq1.dat*



The detection of the completion of this new file is based on **inotify**

inotify is a Linux kernel feature that monitors file systems and immediately alerts an attentive application to relevant events, in our case a "write and close" event (*file opened for writing was closed)*.

It is used within a bash script, running in background.

This mechanism avoids to scan the storage area every *n* seconds to look for new files.

As soon as Inotify detects the file, **3** actions are triggered:

1) the file is copied in the storage area of the farm ★

There is one directory for each run→ if a new run starts, a new directory is created.
The file is copied in its final destination

2) One entry is written in the /todo directory *todo/run1014-seq2.dat*

3) The log file to record all the transfers all_logs/fromEVB/alltransfer.log is updated by appending the following line:

0 2016-10-27 13:55:04 2016-10-27 13:55:04 1023M 2016-10-27 13:55 /home/prod/storage/input/run1014/run1014-seq2.dat

Time duration of the copy from online machine to storage area ★
Time when the copy ended
Time when the copy started
Time when the file was written by the daq

→ A single log file is continuously updated during the data taking

These informations are critical, so some redundancy has been foreseen →the same line is also written in the file /all_logs/fromEVB/runs/run1014.log → one log file for each run

```
 more ../all_logs/fromEVB/runs/run1014.log
0 2016-10-27 13:54:49 2016-10-27 13:54:49 1023M 2016-10-27 13:54 /home/prod/storage/input/run1014/run1014-seq1.dat
0 2016-10-27 13:55:04 2016-10-27 13:55:04 1023M 2016-10-27 13:55 /home/prod/storage/input/run1014/run1014-seq2.dat
0 2016-10-27 13:55:18 2016-10-27 13:55:18 1023M 2016-10-27 13:55 /home/prod/storage/input/run1014/run1014-seq3.dat
0 2016-10-27 13:55:33 2016-10-27 13:55:33 1023M 2016-10-27 13:55 /home/prod/storage/input/run1014/run1014-seq4.dat
0 2016-10-27 13:55:47 2016-10-27 13:55:47 1023M 2016-10-27 13:55 /home/prod/storage/input/run1014/run1014-seq5.dat
0 2016-10-27 13:56:02 2016-10-27 13:56:02 1023M 2016-10-27 13:56 /home/prod/storage/input/run1014/run1014-seq6.dat
0 2016-10-27 13:56:17 2016-10-27 13:56:17 1023M 2016-10-27 13:56 /home/prod/storage/input/run1014/run1014-seq7.dat
0 2016-10-27 13:56:31 2016-10-27 13:56:31 1023M 2016-10-27 13:56 /home/prod/storage/input/run1014/run1014-seq8.dat
0 2016-10-27 13:56:46 2016-10-27 13:56:46 1023M 2016-10-27 13:56 /home/prod/storage/input/run1014/run1014-seq9.dat
0 2016-10-27 13:57:01 2016-10-27 13:57:01 1023M 2016-10-27 13:56 /home/prod/storage/input/run1014/run1014-seq10.dat
```

To be done:

These log files have to be regularly copied in a public area at CERN to be accessible
by everyone.

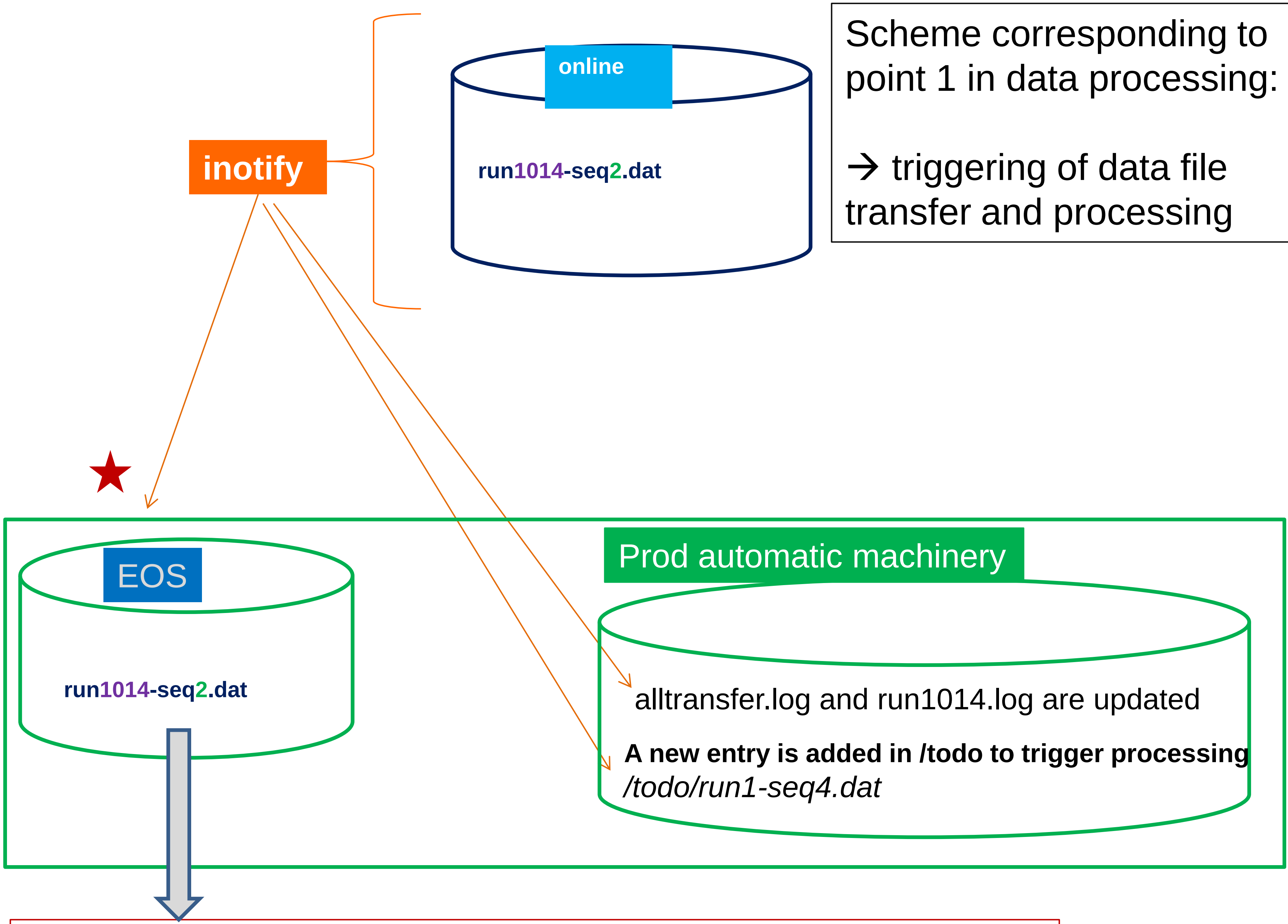
They contain relevant information on runs data flow: number of sequences, time at which each data sequence was copied on local EOS ….

These info are also needed to make some performances studies on the system and to monitor the smoothness of the data flow

The most natural way to treat these files is to put them in a database → to be designed

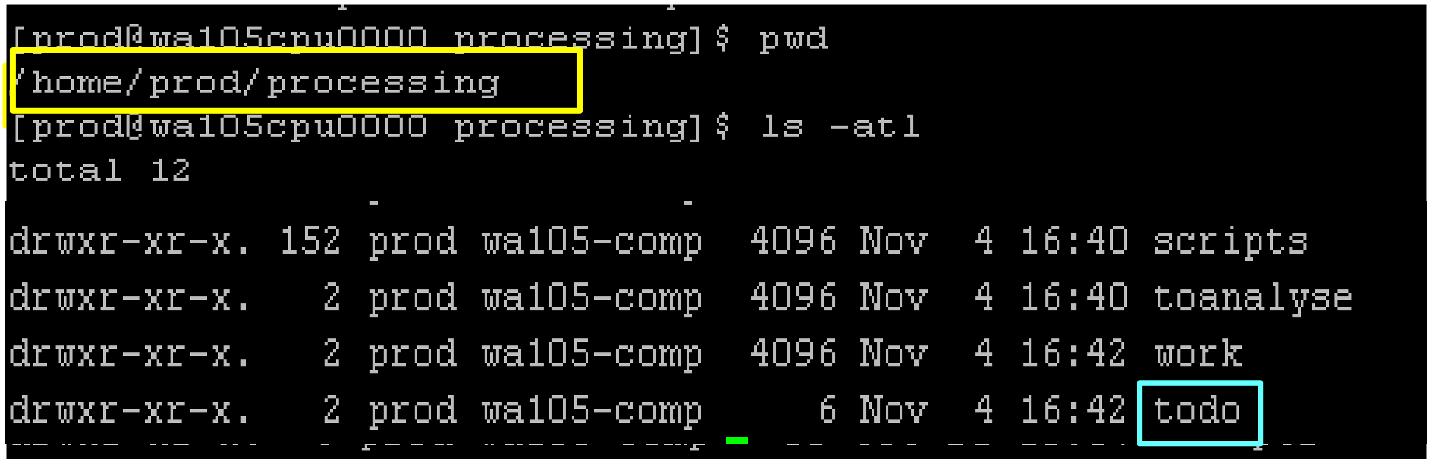
Scheme corresponding to point 1 in data processing:

→ triggering of data file transfer and processing

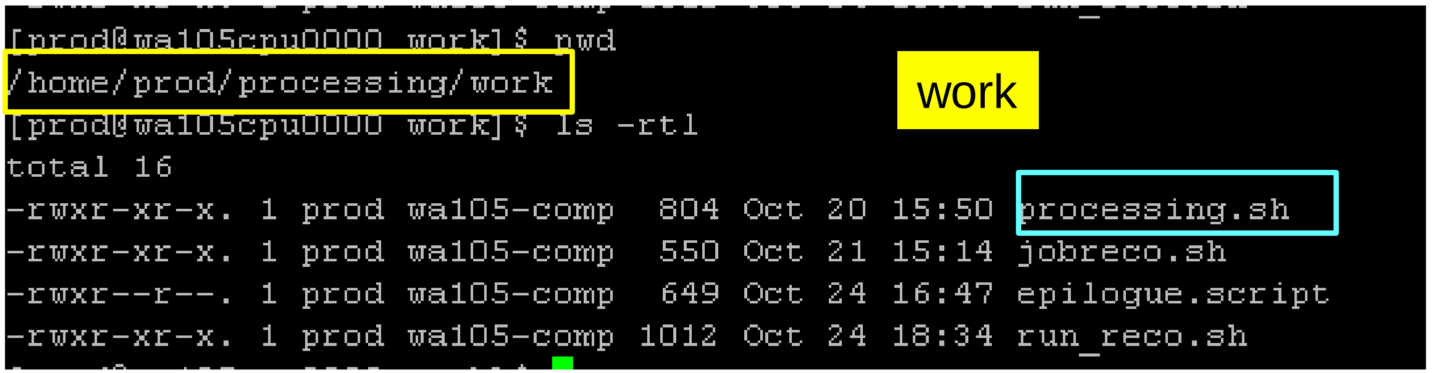

online
run1014-seq2.dat
inotify
EOS
run1014-seq2.dat
Prod automatic machinery
alltransfer.log and run1014.log are updated
A new entry is added in /todo to trigger processing
/todo/run1-seq4.dat

The file has to be processed as soon as it becomes available on eos

In order to handle the processing the manager script processing.sh is periodically executed from the crontab: it reads entries to be processed from /todo dir, it creates a processing script for each file and submits it to the batch system where the load is automatically balanced among workers ★



```
[prod@wa105cpu0000 processing]$ pwd
/home/prod/processing
[prod@wa105cpu0000 processing]$ ls -atl
total 12
drwxr-xr-x. 152 prod wa105-comp  4096 Nov  4 16:40 scripts
drwxr-xr-x.   2 prod wa105-comp  4096 Nov  4 16:40 toanalyse
drwxr-xr-x.   2 prod wa105-comp  4096 Nov  4 16:42 work
drwxr-xr-x.   2 prod wa105-comp     6 Nov  4 16:42 todo
```

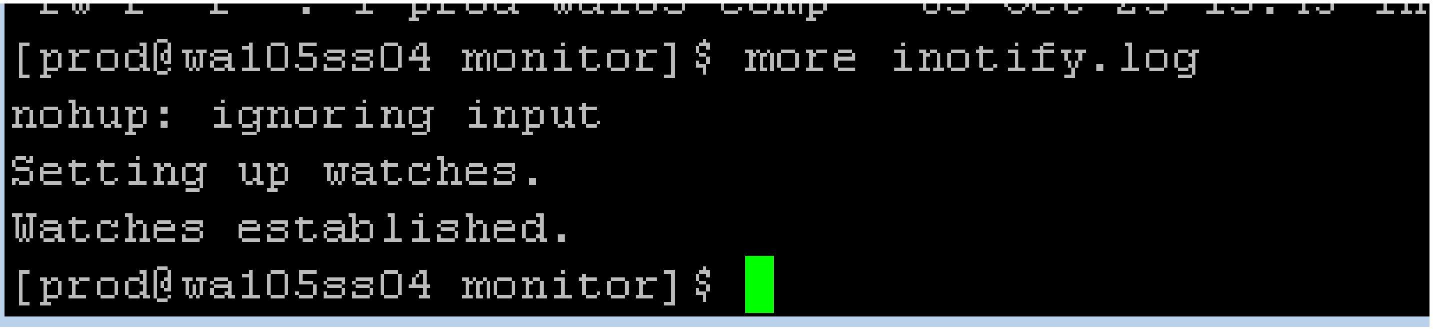


```
[prod@wa105cpu0000 work]$ pwd
/home/prod/processing/work
[prod@wa105cpu0000 work]$ ls -rtl
total 16
-rwxr-xr-x. 1 prod wa105-comp  804 Oct 20 15:50 processing.sh
-rwxr-xr-x. 1 prod wa105-comp  550 Oct 21 15:14 jobreco.sh
-rwxr--r--. 1 prod wa105-comp  649 Oct 24 16:47 epilogue.script
-rwxr-xr-x. 1 prod wa105-comp 1012 Oct 24 18:34 run_reco.sh
```

- All the generated processing scripts are also archived in processing/scripts/runID/…
- The batch processing logs are saved in all_logs/batch/runID/…

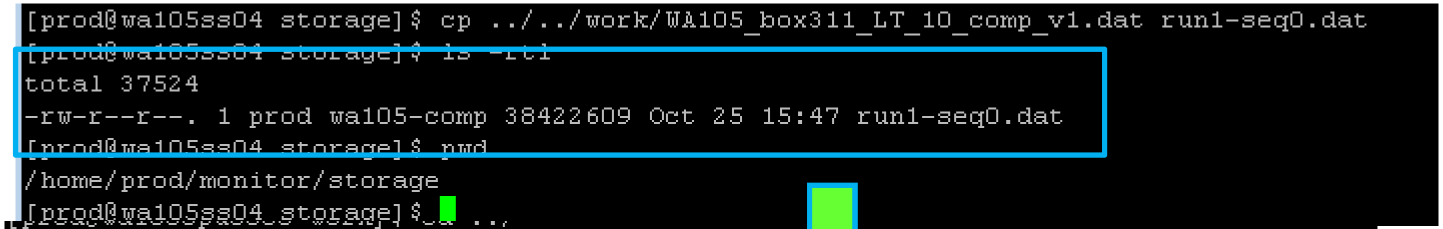
In the following slides all these steps are shown in details

On the online machine, the monitoring script is constantly running:

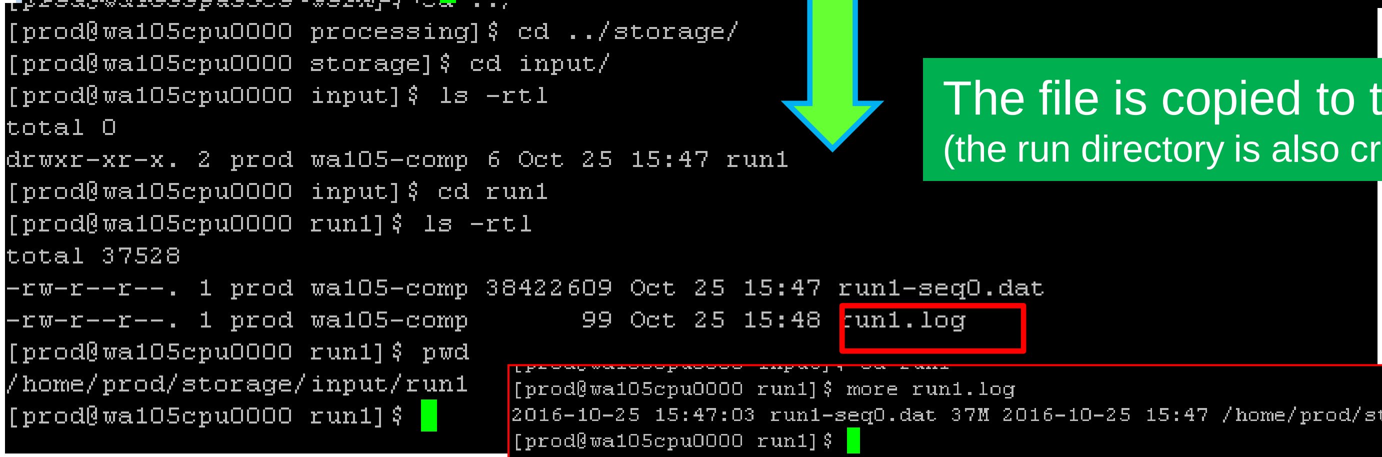


```
[prod@wa105ss04 monitor]$ more inotify.log
nohup: ignoring input
Setting up watches.
Watches established.
[prod@wa105ss04 monitor]$
```

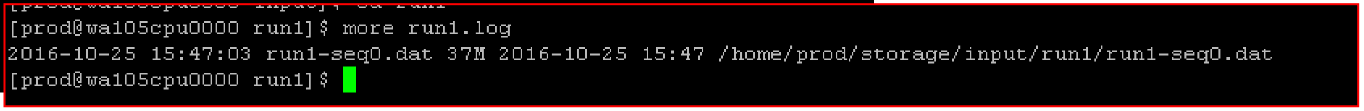
A new data file is now created and the inotify condition is triggered:



```
[prod@wa105ss04 storage]$ cp ../../work/WA105_box311_LT_10_comp_v1.dat run1-seq0.dat
[prod@wa105ss04 storage]$ ls -rtl
total 37524
-rw-r--r--. 1 prod wa105-comp 38422609 Oct 25 15:47 run1-seq0.dat
[prod@wa105ss04 storage]$ pwd
/home/prod/monitor/storage
[prod@wa105ss04 storage]$
```



```
[prod@wa105cpu0000 processing]$ cd ../storage/
[prod@wa105cpu0000 storage]$ cd input/
[prod@wa105cpu0000 input]$ ls -rtl
total 0
drwxr-xr-x. 2 prod wa105-comp 6 Oct 25 15:47 run1
[prod@wa105cpu0000 input]$ cd run1
[prod@wa105cpu0000 run1]$ ls -rtl
total 37528
-rw-r--r--. 1 prod wa105-comp 38422609 Oct 25 15:47 run1-seq0.dat
-rw-r--r--. 1 prod wa105-comp       99 Oct 25 15:48 run1.log
[prod@wa105cpu0000 run1]$ pwd
/home/prod/storage/input/run1
[prod@wa105cpu0000 run1]$
```

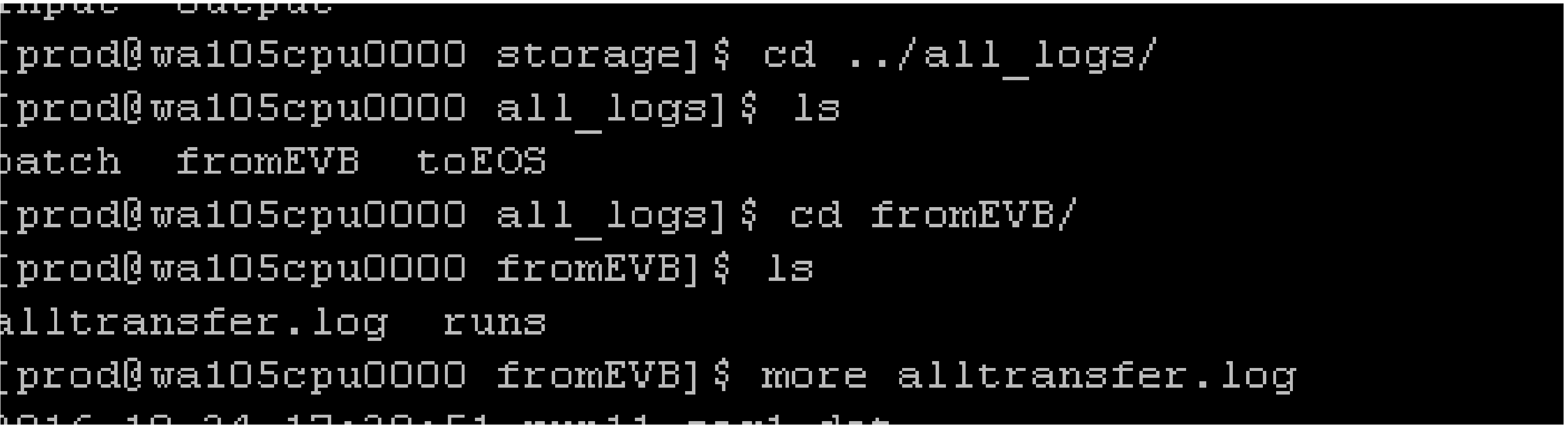


```
[prod@wa105cpu0000 run1]$ more run1.log
2016-10-25 15:47:03 run1-seq0.dat 37M 2016-10-25 15:47 /home/prod/storage/input/run1/run1-seq0.dat
[prod@wa105cpu0000 run1]$
```

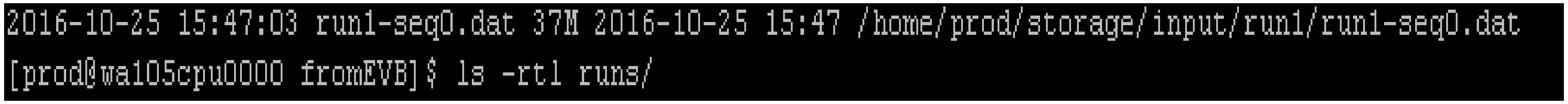
The file is copied to the storage
(the run directory is also created)

★

The file transfer action is recorded in the log files:

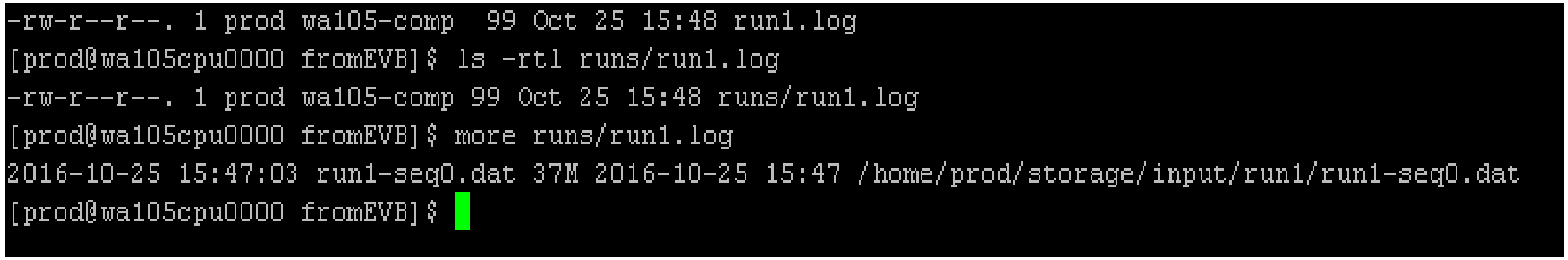


```
[prod@wa105cpu0000 storage]$ cd ../all_logs/
[prod@wa105cpu0000 all_logs]$ ls
batch  fromEVB  toEOS
[prod@wa105cpu0000 all_logs]$ cd fromEVB/
[prod@wa105cpu0000 fromEVB]$ ls
alltransfer.log  runs
[prod@wa105cpu0000 fromEVB]$ more alltransfer.log
```

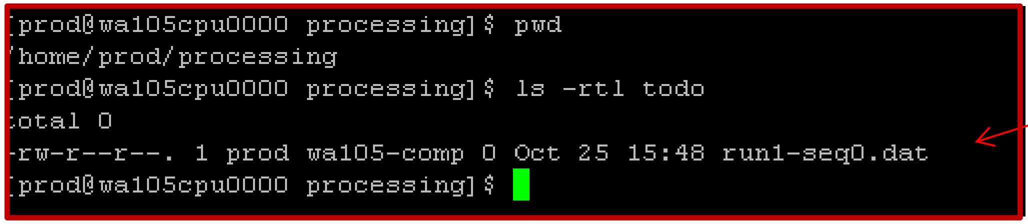


```
2016-10-25 15:47:03 run1-seq0.dat 37M 2016-10-25 15:47 /home/prod/storage/input/run1/run1-seq0.dat
[prod@wa105cpu0000 fromEVB]$ ls -rtl runs/
```

The same information is also logged in the run logfile:

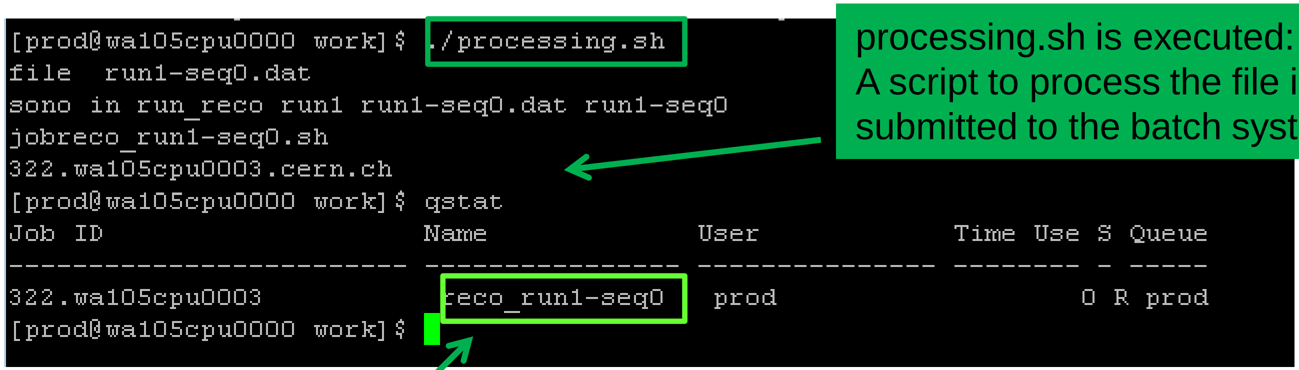


```
-rw-r--r--. 1 prod wa105-comp  99 Oct 25 15:48 run1.log
[prod@wa105cpu0000 fromEVB]$ ls -rtl runs/run1.log
-rw-r--r--. 1 prod wa105-comp 99 Oct 25 15:48 runs/run1.log
[prod@wa105cpu0000 fromEVB]$ more runs/run1.log
2016-10-25 15:47:03 run1-seq0.dat 37M 2016-10-25 15:47 /home/prod/storage/input/run1/run1-seq0.dat
[prod@wa105cpu0000 fromEVB]$ 
```



```
[prod@wa105cpu0000 processing]$ pwd
/home/prod/processing
[prod@wa105cpu0000 processing]$ ls -rtl todo
total 0
-rw-r--r--. 1 prod wa105-comp 0 Oct 25 15:48 run1-seq0.dat
[prod@wa105cpu0000 processing]$
```

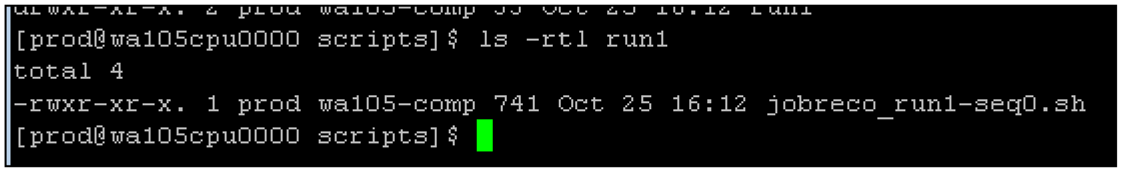
And a line is added in todo directory



```
[prod@wa105cpu0000 work]$ ./processing.sh
file  run1-seq0.dat
sono in run_reco run1 run1-seq0.dat run1-seq0
jobreco_run1-seq0.sh
322.wa105cpu0003.cern.ch
[prod@wa105cpu0000 work]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
322.wa105cpu0003           reco_run1-seq0   prod                   0 R prod
[prod@wa105cpu0000 work]$
```

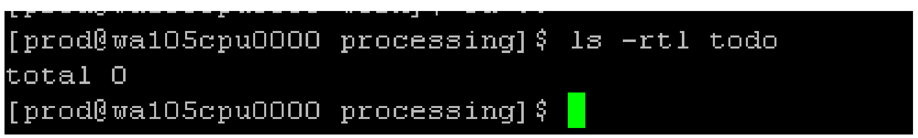
processing.sh is executed:
A script to process the file is generated and it is submitted to the batch system

The job is named reco_filename



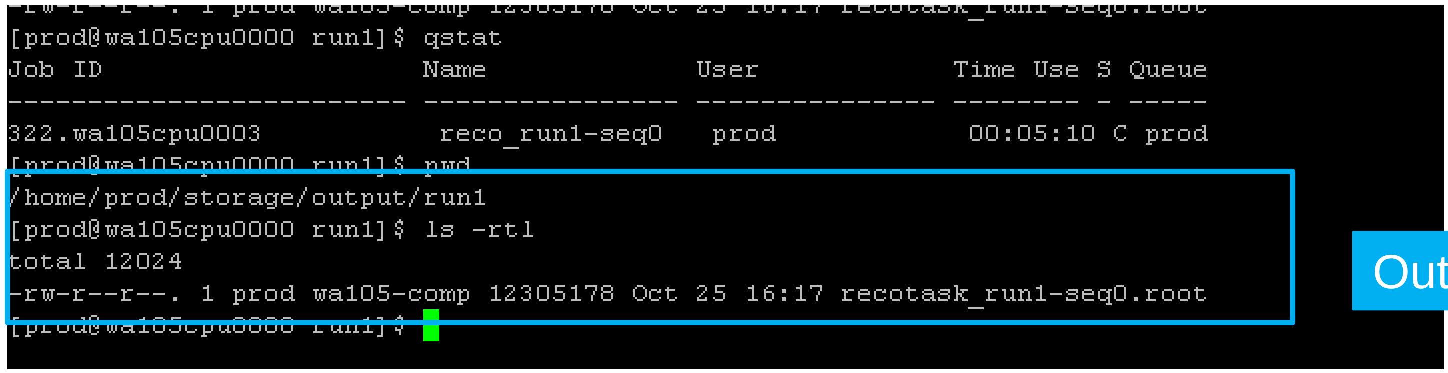
```
drwxr-xr-x. 2 prod wa105-comp 33 Oct 25 16:12 run1
[prod@wa105cpu0000 scripts]$ ls -rtl run1
total 4
-rwxr-xr-x. 1 prod wa105-comp 741 Oct 25 16:12 jobreco_run1-seq0.sh
[prod@wa105cpu0000 scripts]$
```

and the line is removed for the todo directory

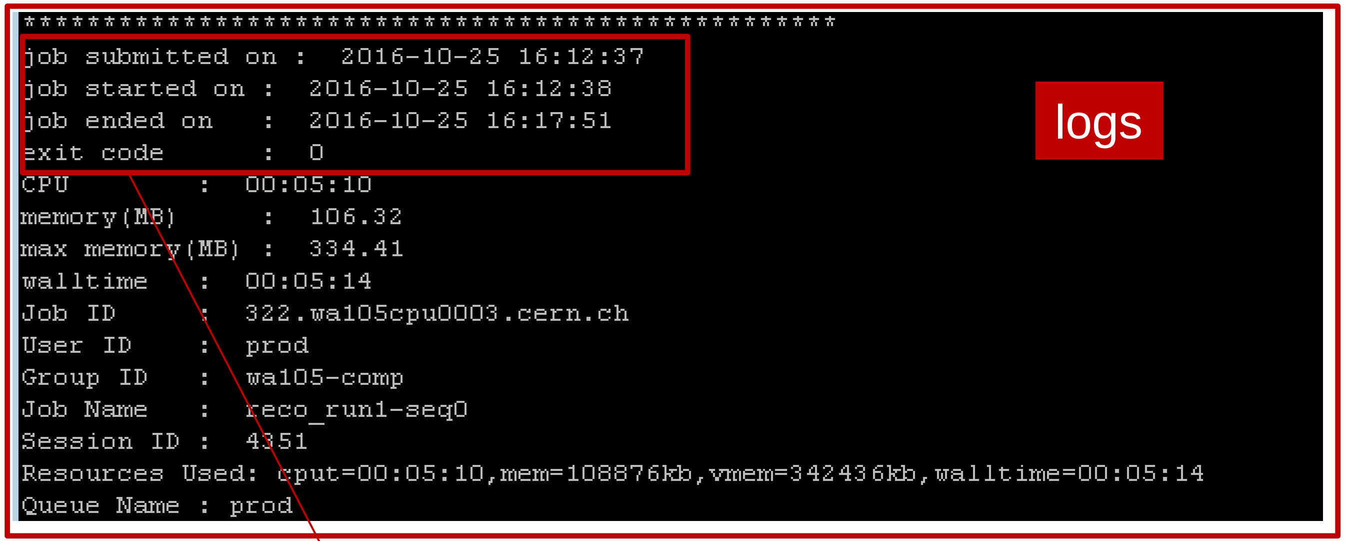


```
[illegible]
[prod@wa105cpu0000 processing]$ ls -rtl todo
total 0
[prod@wa105cpu0000 processing]$
```

Batch results



```
-rw-r--r--. 1 prod wa105-comp 12305178 Oct 25 16:17 recotask_run1-seq0.root
[prod@wa105cpu0000 run1]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
322.wa105cpu0003           reco_run1-seq0   prod            00:05:10 C prod
[prod@wa105cpu0000 run1]$ pwd
/home/prod/storage/output/run1
[prod@wa105cpu0000 run1]$ ls -rtl
total 12024
-rw-r--r--. 1 prod wa105-comp 12305178 Oct 25 16:17 recotask_run1-seq0.root
[prod@wa105cpu0000 run1]$
```



```
**************************************************
job submitted on :  2016-10-25 16:12:37
job started on :  2016-10-25 16:12:38
job ended on    :  2016-10-25 16:17:51
exit code       :  0
CPU          :  00:05:10
memory(MB)      :  106.32
max memory(MB) :  334.41
walltime     :  00:05:14
Job ID       :  322.wa105cpu0003.cern.ch
User ID      :  prod
Group ID     :  wa105-comp
Job Name     :  reco_run1-seq0
Session ID :  4351
Resources Used: cput=00:05:10,mem=108876kb,vmem=342436kb,walltime=00:05:14
Queue Name : prod
```

These info can be combined in the global data flow logging to the ones related to the transfer of the data files to the farm storage/ CERN EOS

Treatment of job failure:

- At the moment there is no automatic resubmission in case of job failure.
- The failure is notified by sending a mail to prod
- It is reasonable to start like this, and learn during the data taking which is the amount of jobs for which the reconstruction fails and understand for which reasons in order to tune a recovery procedure
- From this first experience we can decide if and in which cases automatic resubmission is needed

Tests of the system

- To validate this setting-up, some tests have been performed.
- The goal was to test the synchronization mechanism between the online machine and the production farm, the automatic processing of the binary files and the batch system performances
- These tests will have to be repeated once the local EOS configuration will be finished ★

Test #1 :
A binary file **(compressed)** is produced **every 300 seconds** (corresponding event trigger rate~1Hz) by the online machine,
Total number of files generated and processed during the test: 50
The test lasted ~ 4hours and 10 min

1st job submitted on : 2016-10-26 26 11:01:00
Last job ended on : 2016-10-26 26 15:11:11

Test #2 :
A binary file **(not compressed)** is produced **every 300 seconds** by the online,
Total number of files generated and processed during the test: 25
The test lasted ~ 2 hours and 5 min

1st job submitted on : 2016-10-28 09:21:22
Last job ended on : 2016-10-28 11:25:42

Note: data compression is an aspect which has been implemented in the DAQ and it is under test in view of the 6x6x6 where it will be essential but it is not needed in the 3x1x1 data flow

→ No problems during jobs processing: as soon as one raw data file is produced by the online is "immediately" treated the queues for batch processing are not saturated at all,
A job submitted to the batch system is immediately processed

To further validate the system, two different tests have been performed:

Test #3: A binary file **(compressed)** is produced **every 10 seconds** (event trigger rate~30Hz) by the online,
Total number of files generated and processed during the test: 500
The test lasted ~ 1hours and 30 min

1st job submitted on : 2016-11-04 12:45:19
last job ended on: 2016-11-04 14:19:58

Test #4: A binary file **(not compressed)** is produced **every 10 seconds** (event trigger rate~30Hz) by the online,
Total number of files generated and processed during the test: 500
The test lasted ~ 2hours and 5 min

1st job submitted on : 2016-11-04 14:39:44
last job ended on: 2016-11-04 16:45:13

→ In both cases the batch processing ended smoothly, and once a job is submitted to the batch system is immediately processed

The time duration of the two test is different: this is due to how the files were "generated" on the online machine, this is related neither to the synchronization mechanisms between online machine and processing farm, nor to the batch system
(these fake DAQ files are generated by multiple copies from the same data file, this process is more efficient with small files (compressed data file) than with the standard 1 GB files

- In the different tests, all the jobs ended correctly.

- All the available workers have executed at least one job: results have been checked also with Denis, who installed and configured the batch system. This has allowed to find and fix some (minor) issues in the batch system configuration.

- Some numbers:
  RAM memory needed by each job ~ 330MB

  Time elapsed between the start and the end of the execution (335 events/file)
  compressed data input file: ~ 9 minutes
  not compressed data input file: ~ 4 minutes

- With Denis we have decided that the input file to be processed is not copied to the $TMPDIR of the worker, and the output file is also directly written in the storage area → this avoids file transfers, the network bandwidth and reduces metadata handling

- These tests have been done more than once to check in time the stability of the system, and in general they were repeated every time a modification in the configuration of the online processing/storage system was introduced

- All the root files have been systematically opened for reading and the number of entries checked

- The execution logs contain time information on the processing of the files→ these can be merged with the ones related to file transfer, to build the complete ”story” of the file

Purity and gain analysis (point 2)

2) Results from reconstruction (root files) are also stored in the storage area and analyzed to evaluate purity and gain, to monitor the behavior of the detector in time (*online analysis*)

- The purity and gain analysis is performed starting for root file obtained from data processing (the principle and the tools have been presented extensively at the past SB meetings)
- Several root files (3 or 4) have to be combined in order to gather sufficient statistics to produce significant results ★
- The scripts needed to execute the analysis in an automatic way are under development, but what is important is that all the required software is available : it has been presented and explained during the Science Board meeting of July 6th and it is also available on the svn https://indico.fnal.gov/conferenceDisplay.py?confId=12481
- The gain analysis requires as input the results obtained from purity measurement (these results are written in an ascii file) Slavic has modified the code for gain measurement to read this external file
- Results for purity and gain can be used during the shifts to monitor the detector behavior: Slavic is developing the code to monitor them

LEM gain

The method was presented at SB July/06

[svn] / WA105Soft / src / oanacrpgain.cc

View of /WA105Soft/src/oanacrpgain.cc

- The program segments CRP into 50x50 cm2 areas
- The dQ/ds from tracks crossing each area are stored
- The mean / (or truncated mean) is calculated after a given number of events is processed

Inputs:

1. File with reconstructed 2D tracks
2. File with summary of the purity analysis: need to correct the charge attenuation

Output:

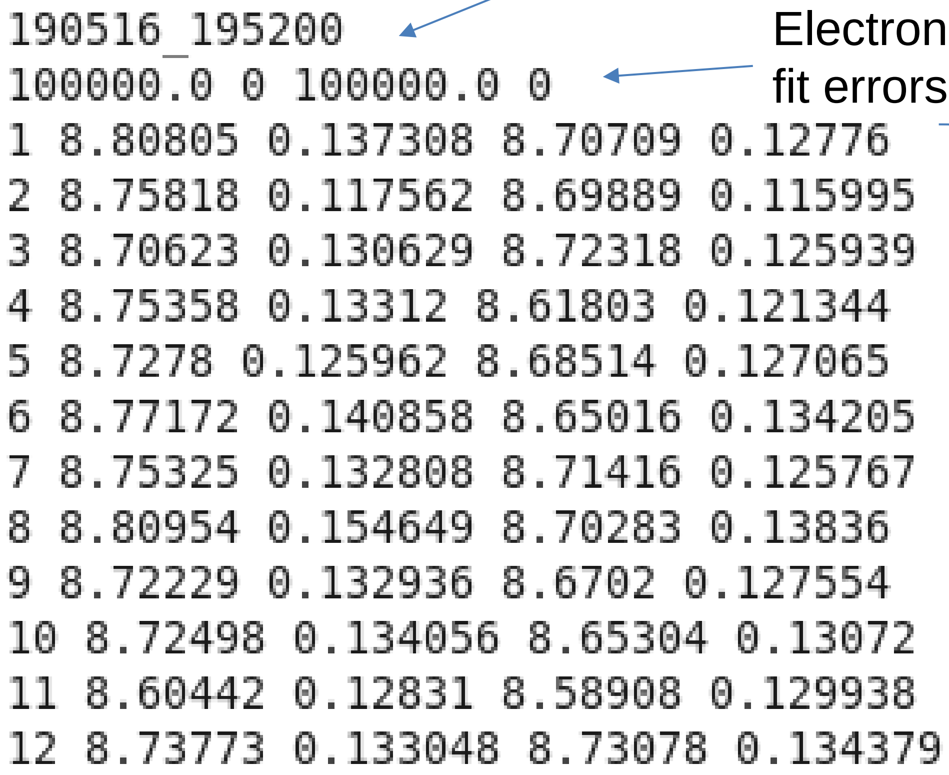
1. If selected, can append to the purity analysis file
2. Optionally can produce ROOT file with 2D histogram of the segmented area

SG

Example summary file

Time stamp: DDMMYY_HHMMSS

Electron lifetime (from view 0, view 1) in us and fit errors. (some dummy values here for inf)



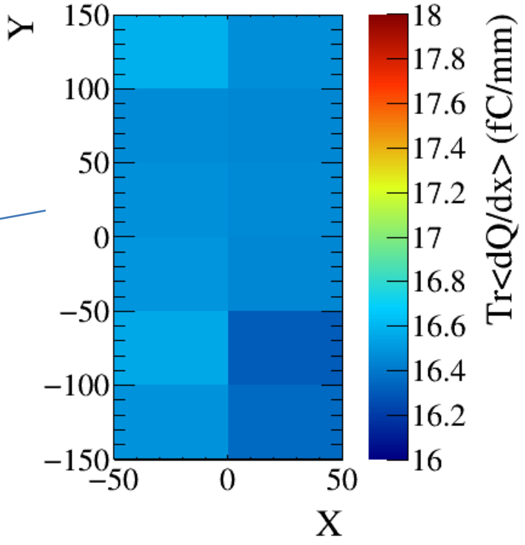
```
190516_195200
100000.0 0 100000.0 0
1 8.80805 0.137308 8.70709 0.12776
2 8.75818 0.117562 8.69889 0.115995
3 8.70623 0.130629 8.72318 0.125939
4 8.75358 0.13312 8.61803 0.121344
5 8.7278 0.125962 8.68514 0.127065
6 8.77172 0.140858 8.65016 0.134205
7 8.75325 0.132808 8.71416 0.125767
8 8.80954 0.154649 8.70283 0.13836
9 8.72229 0.132936 8.6702 0.127554
10 8.72498 0.134056 8.65304 0.13072
11 8.60442 0.12831 8.58908 0.129938
12 8.73773 0.133048 8.73078 0.134379
```

<dQ/ds> in fC/cm obtained in each 50x50 cm2 from both views
But LEM numbering needs to be adapted to the 3x1x1 convention

SG

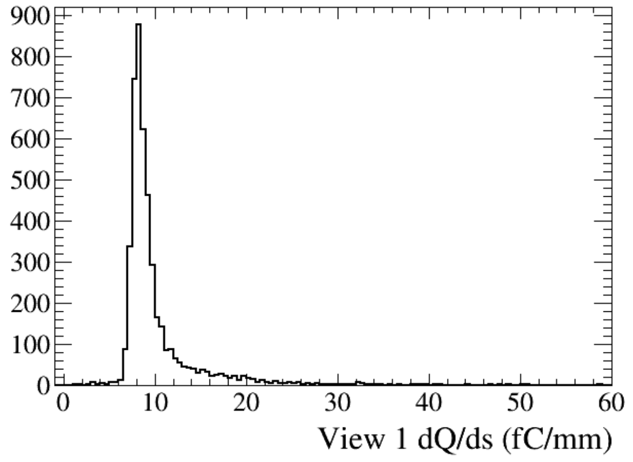
Optional ROOT output file

2D histo showing sum of <dQ/ds> from two views



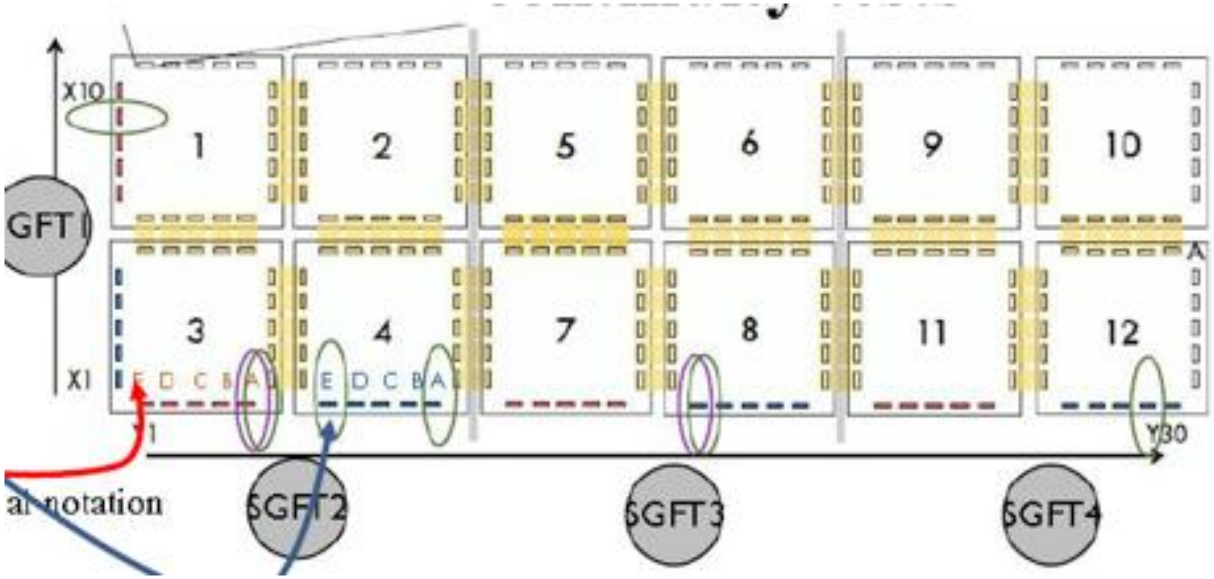
```
KEY: TH2F     crp3x1x1_crpmap;1
KEY: TH1F     crp3x1x1_view0_bin0;1
KEY: TH1F     crp3x1x1_view1_bin0;1
KEY: TH1F     crp3x1x1_view0_bin1;1
KEY: TH1F     crp3x1x1_view1_bin1;1
KEY: TH1F     crp3x1x1_view0_bin2;1
KEY: TH1F     crp3x1x1_view1_bin2;1
KEY: TH1F     crp3x1x1_view0_bin3;1
KEY: TH1F     crp3x1x1_view1_bin3;1
KEY: TH1F     crp3x1x1_view0_bin4;1
KEY: TH1F     crp3x1x1_view1_bin4;1
KEY: TH1F     crp3x1x1_view0_bin5;1
KEY: TH1F     crp3x1x1_view1_bin5;1
KEY: TH1F     crp3x1x1_view0_bin6;1
KEY: TH1F     crp3x1x1_view1_bin6;1
KEY: TH1F     crp3x1x1_view0_bin7;1
KEY: TH1F     crp3x1x1_view1_bin7;1
KEY: TH1F     crp3x1x1_view0_bin8;1
KEY: TH1F     crp3x1x1_view1_bin8;1
KEY: TH1F     crp3x1x1_view0_bin9;1
KEY: TH1F     crp3x1x1_view1_bin9;1
KEY: TH1F     crp3x1x1_view0_bin10;1
KEY: TH1F     crp3x1x1_view1_bin10;1
KEY: TH1F     crp3x1x1_view0_bin11;1
KEY: TH1F     crp3x1x1_view1_bin11;1
```

dQ/ds distribution in each spatial bin



SG

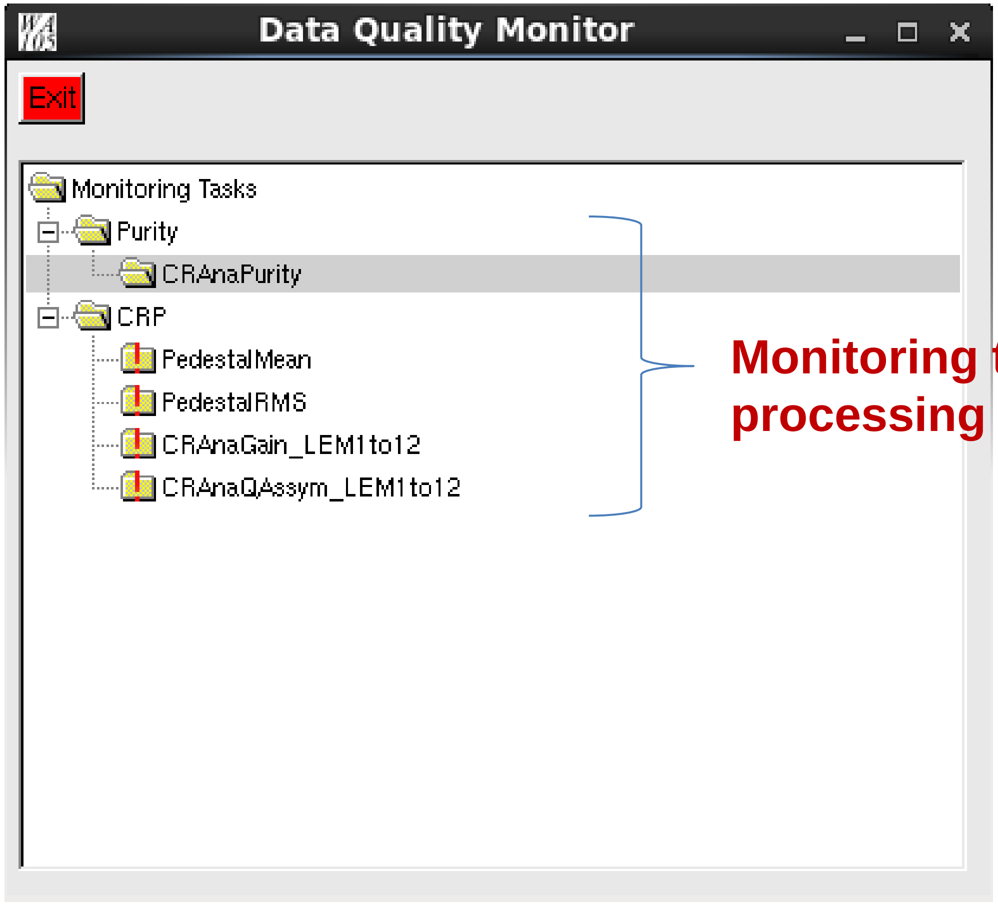
Note on the LEM numbering



The convection for the LEM numbering is rather awkward
In the future it would be much better to continuously increment LEM number along x /y with wrap around to the next row / column (like binning in 2D histogram)

SG

GUI version 0

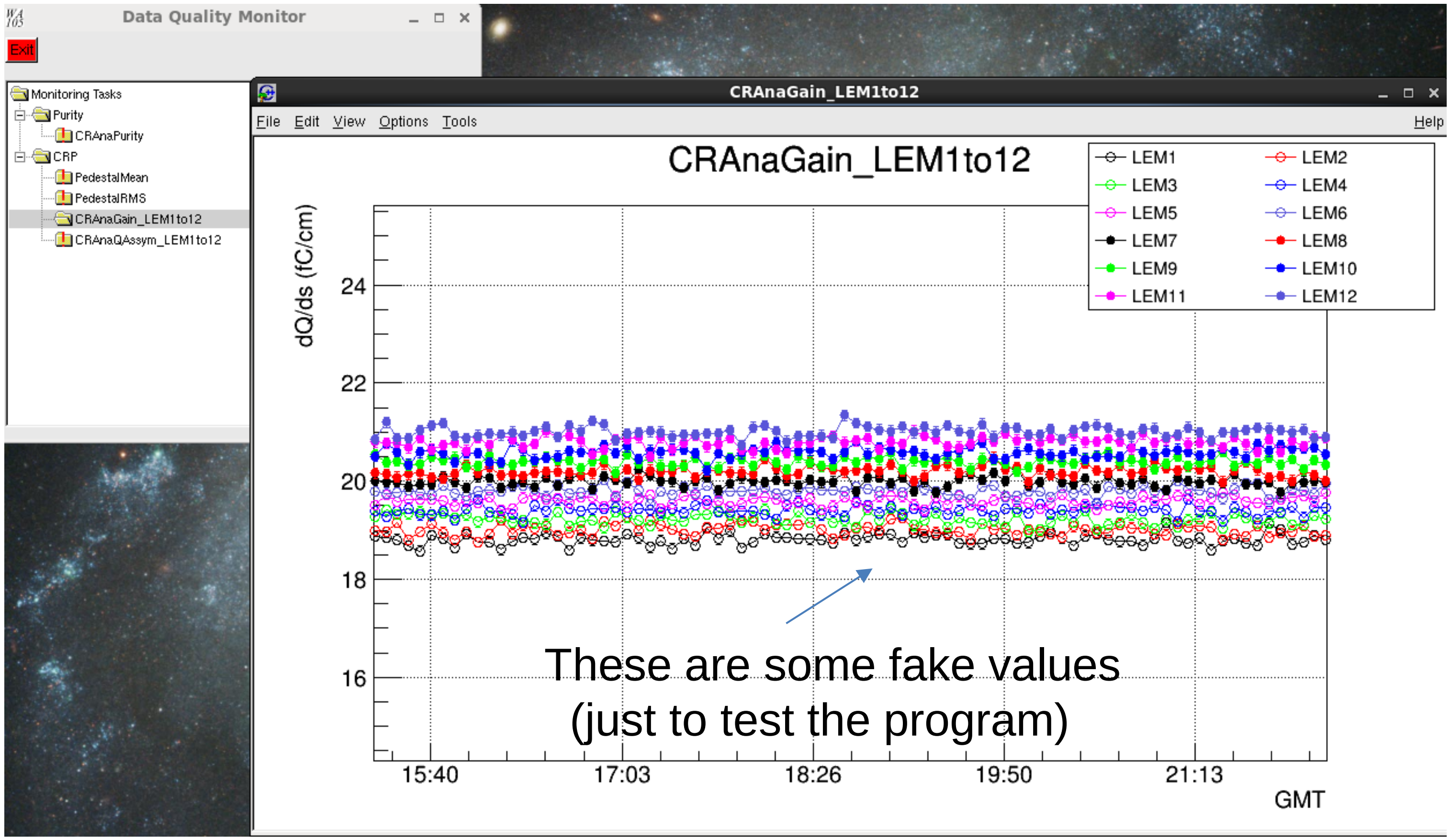


```
Monitoring Tasks
  Purity
    CRAnaPurity
  CRP
    PedestalMean
    PedestalRMS
    CRAnaGain_LEM1to12
    CRAnaQAssym_LEM1to12
```

Monitoring tasks from online processing outputs

SG

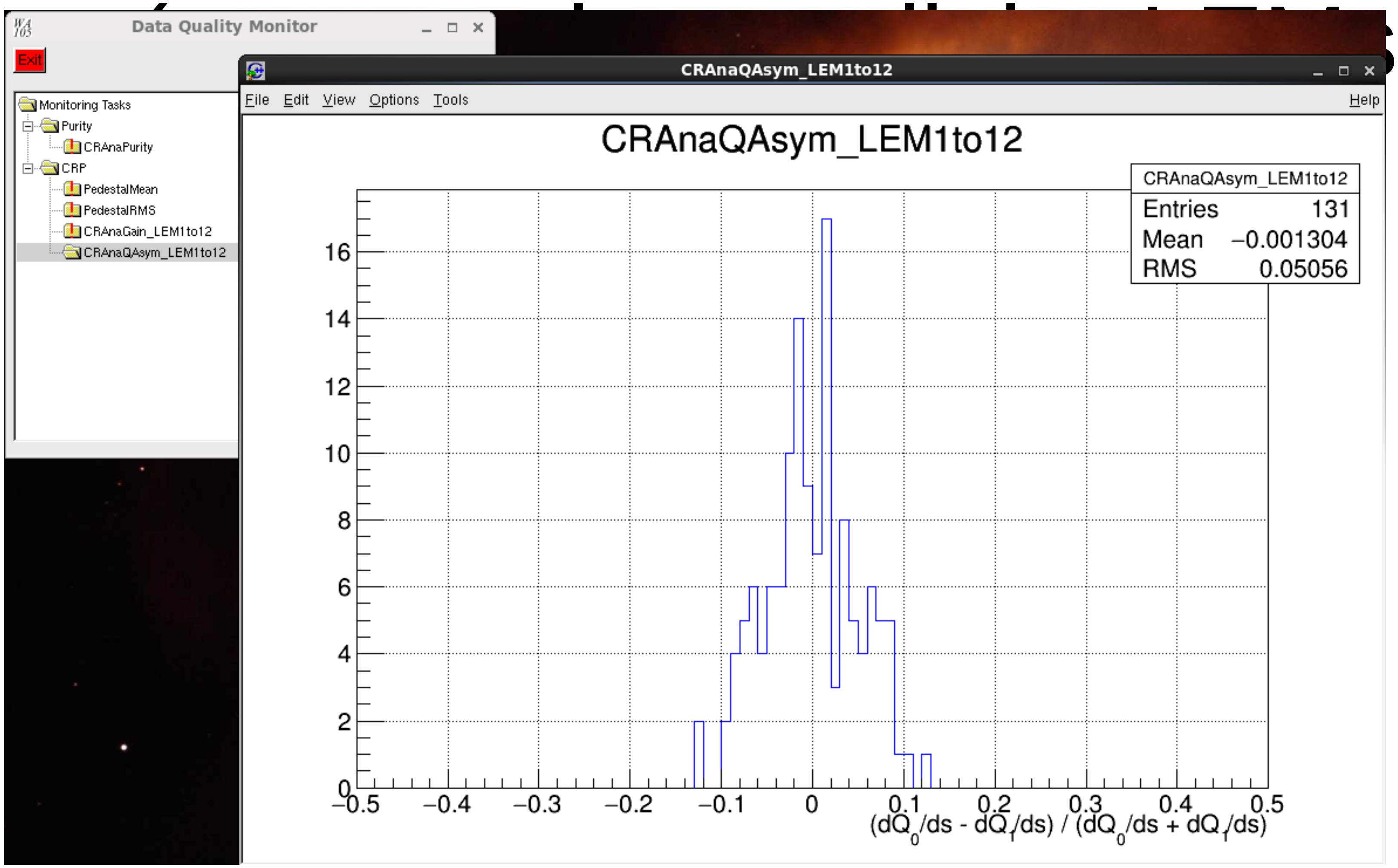
LEM gain monitoring



These are some fake values
(just to test the program)

SG

Charge asymmetry between views
([illegible]s)



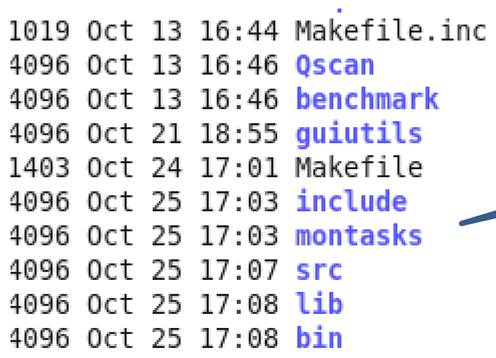
CRAnaQAsym_LEM1to12

Entries	131
Mean	−0.001304
RMS	0.05056

SG

Adding more monitoring tasks

Here it is shown how new distributions can be included in the display:



```
1019 Oct 13 16:44 Makefile.inc
4096 Oct 13 16:46 Qscan
4096 Oct 13 16:46 benchmark
4096 Oct 21 18:55 guiutils
1403 Oct 24 17:01 Makefile
4096 Oct 25 17:03 include
4096 Oct 25 17:03 montasks
4096 Oct 25 17:07 src
4096 Oct 25 17:08 lib
4096 Oct 25 17:08 bin
```

Add code for a specific task here

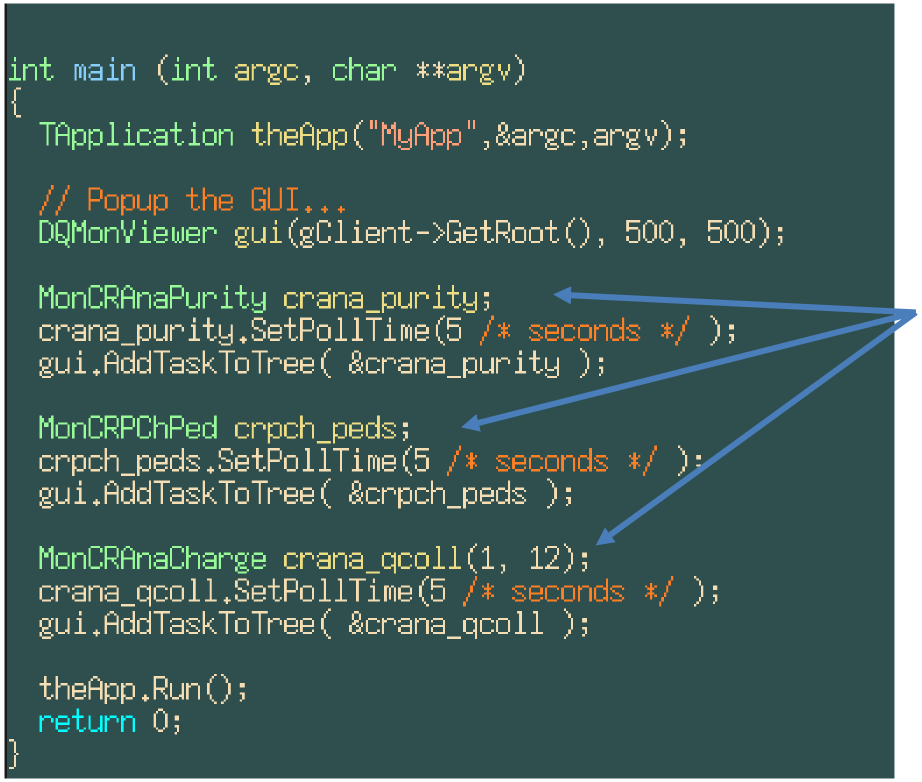
A given task should be derived from MonTask object
Need to implement:

- The names of different items to appear in the tree
- A function on how to read a given file with results
- A function on how to draw the data

SG

Adding more monitoring task

- Add the new task to the gui “src/dqmon.cc”



```
int main (int argc, char **argv)
{
  TApplication theApp("MyApp",&argc,argv);

  // Popup the GUI...
  DQMonViewer gui(gClient->GetRoot(), 500, 500);

  MonCRAnaPurity crana_purity;
  crana_purity.SetPollTime(5 /* seconds */ );
  gui.AddTaskToTree( &crana_purity );

  MonCRPChPed crpch_peds;
  crpch_peds.SetPollTime(5 /* seconds */ );
  gui.AddTaskToTree( &crpch_peds );

  MonCRAnaCharge crana_qcoll(1, 12);
  crana_qcoll.SetPollTime(5 /* seconds */ );
  gui.AddTaskToTree( &crana_qcoll );

  theApp.Run();
  return 0;
}
```

This are the tasks shown so far

The polling time sets the frequency with which the data summary directory is checked for new results (here 5 sec were used for testing and should be more reasonable in reality)

SG

Transfer of raw data files to central EOS (point 3)

3) The binary raw data files are also copied to the central CERN EOS, where they are available to the users for offline analysis

- As discussed with IT people, data transfer to the central EOS (and CASTOR) should happen by clustering the events in files having dimensions of a few gigabytes: ideally 5GB, as it will be done for the 6x6x6, for the 3x1x1 we kept to the minimum of 1 GB allowed by the system for a decent performance

→ The standard data size for the 3x1x1 will be 1 GB with copies also to the central CERN EOS,

some tests at 5 GB could be useful for performance studies in view of the 6x6x6 :

- In order to assemble the data in larger files for final archiving on the central EOS the safer and easiest approach is to build a tar archive. BUT:
- The run duration is not fixed, and especially in the first period of the data taking, we can imagine to have short runs (with few events), corresponding to different running conditions.
- So, to build an archive of 5 GB we would have to group several runs. At least in the first part of the data taking it would be better to build the archives by hands, in order to group together runs corresponding to the same type of measurement (eg, runs corresponding to an HV scan) Of course a log file has to be provided with every archive
- Scripts to move files between different EOS instance have already been prepared, but of course will have to be modified to run on the farm, to check the transfer rate and optimize the options chosen for the transfer

This part is not in top priority it will be tested calmly when the situation is stable in order to prepare for the 6x6x6

Conclusions

- The processing of raw data on the online farm has been setup: the needed software is available and the batch system has been carefully tested and benchmarked

- The software needed to perform online analysis (purity+gain) is available, the scripts to automatize its execution are in progress.
  Slavic is working at the monitoring of these results

- For each step in the processing log files are written, with the final goal to store them in a database. This is not urgent, but it is something that has to be foreseen and tested in view of the 6x6x6 operation

- Beyond the monitoring of the 3x1x1 all this work is mostly useful in order to test and prepare for the 6x6x6 and optimize the design of the final farm

TO DO:

- Once EOS is available, all parts of the machinery concerning data access will have to be modified and all tests will be redone and the data storage/network will be benchmarked
- Some work to automatize the execution of the online analysis for purity/gain has to be completed
- The version of the WA105Soft used is not the final one: for sure some changes will be needed once the DAQ program will be in its final version. Once we are confident that the software version is the final one, a new data reconstruction qscan release can be tagged, defined as the "official one" and used for the online and offline processing